

Some Image Processing Algorithms on a RAP with Wider Bus Networks

Shung-Shing Lee , Shi-Jinn Horng , Horng-Ren Tsai and Yu-Hua Lee
National Taiwan Institute of Technology
Department of Electrical Engineering
43, Section 4, Kee-Lung Road, Taipei, Taiwan, R. O. C.
horng@pde.ee.ntit.edu.tw

Abstract

Based on the reconfigurable array of processors with wider bus networks [8], we propose a series of algorithms for image processing. Conventionally, only one bus is connected between two processors but in this machine it has a set of buses. Such a characteristic increases the computation power of this machine greatly. Based on the base- m number system, we first introduce some basic operation algorithms. Then three related applications are derived in constant time; one is the histogram of an image, another is the image segmentation and the other is the image labeling.

Key Words— Image processing, parallel algorithms, entropy, prefix sum, histogram, Segmentation, reconfigurable array of processors, bus network, labeling.

1 Introduction

Owing to the fact that the speed of the electron is limited, the computation power of a single processor cannot be significantly increased. Instead of designing super processors, many researchers have focused their attentions to the parallel processing systems. As we know, the computation power of a parallel processing system cannot be linearly increased in proportion to the number of processors installed to the system. It fully depends on the algorithm designed and the system architecture proposed. The mesh-connected computer (abbreviated to MCC) is one of these famous parallel processing systems [16]. Even though the architecture of the MCC is simple and regular, its architecture is fixed at running time and it is not adequate for global communications. Researchers overcame such draw-

backs by equipping it with a reconfigurable bus system. Several reconfigurable parallel processing systems have been proposed [6, 9, 12, 14, 15, 21, 22, 23, 24]. There are processor arrays with a reconfigurable bus system (abbreviated to PARBS) [23, 24], reconfigurable meshes [14, 15], polymorphic torus networks [9], bus automation [21], reconfigurable array of processors [6], reconfigurable networks [22], and polymorphic processor arrays [12]. Although the system bus of any reconfigurable parallel processing systems is reconfigurable at run time there is still a drawback to these models. That is, the bandwidth of the buses between processors depends on the logarithmic order of the number of the processors to be installed in the system. Consequently, this is no good for those computations that need wider communication bandwidth. Conventionally, researchers solved this problem by installing the system with an extra number of processors. In fact, this problem can be also solved using a wider bus system architecture instead of using more processors. Li and Maresca [10, 13] showed that it would take 20% more silicon area per processor before they could control the local switching between buses at the instruction level. This implies that it would be more efficient to save silicon area by increasing the bus capacity rather than by increasing the processor complexity. Based on such a concept, Lee et al. [8] have proposed a reconfigurable array of processors with wider bus networks to solve Hough transforms efficiently. In this paper we also use it to solve some other interesting image problems such as segmentation and labeling.

The histogram of an image provides a useful information for segmentation and for measuring the textual properties of an image. By definition, the histogram of an image is a one-dimensional array $H[i]$, $i = 0, 1, \dots, G - 1$, such that $H[i] = k$ if k is the number of pixels with grey-level i . In this paper, we use Lee's

[8] result to solve this problem .

After computing the histogram of an image, the segmentation of an image based on the histogram entropy can be progressed. The technique for image segmentation based on the similarity of brightness of image objects is thresholding. To choose a good threshold value of a gray-level image, the image can be segmented into two parts, the object regions and the background. Based on the entropy of the histogram, many authors [7, 19, 20] have used Shannon's concept for thresholding an image. Pun [19, 20] and Kapur et al. [7] defined the entropy of an image by assuming that an image was entirely represented by its gray level histogram only. They then used this entropy to choose a threshold value for the image. Based on this concept, the sequential algorithms for finding a threshold value of an image take $O(G^2)$ time. Here G is the total grey levels of an image. Recently, based on the entropy of the histogram, Cinque et al. [2] proposed some algorithms for image thresholding in $O(\log G)$ time on a pyramidal machine with a $G \times G$ base processors. We present a parallel algorithm for the same problem on the reconfigurable array of processors with wider bus networks in constant time using $G^{2+1/c}$ processors.

For segmentation, we first transform the gray-level image into a binary image. Then, we propose an image labeling algorithm for this binary image. The image labeling problem is to assign a unique number to each connected component (object) of an image. Kao et al. [5] and Alnuweiri [1] have proposed constant time algorithms for the image labeling problem on reconfigurable parallel processing systems. Under the RAPWBN model, we also can solve this problem in constant time.

After labeling an image, each labeled object can be recognized or understood by calculating its characteristics.

The rest of this paper is organized as follows. The reconfigurable array of processors with wider bus networks and some basic operations are described in Sections 2 and 3. Section 4 introduces histogram algorithm. Section 5 shows a segmentation algorithm based on the histogram entropy of an image. An image labeling algorithm is presented in Section 6. Finally, some concluding remarks are also included in the last section.

2 The computation model and Notations

A 1-dimensional (1-D) reconfigurable array of processors with wider bus networks (abbreviated to RAPWBN) [8] of size N contains N processors to be embedded with a reconfigurable bus system. Each processor

is identified by a unique index denoted as P_j , where $0 \leq j < N$. The reconfigurable bus system consists of an M -row and N -column network array and the bandwidth of each bus of each network is assumed to be $O(N^{1/c})$ -bit, where N is the number of processors and c is a constant for $c \geq 1$. Usually, assume $O(N^{1/c}) = m$. The M -row and N -column bus networks have $2MN$ ports denoted by $-S_{i,j}, +S_{i,j}$ and each port has m -bit bus connection switches denoted by $-S_{i,j}(k), +S_{i,j}(k)$, for $0 \leq i < M, 0 \leq j < N$ and $0 \leq k < m$. The i^{th} row bus is constructed by connecting the j^{th} -column's port switch $+S_{i,j}$ to the $(j+1)^{th}$ -column's port switch $-S_{i,j+1}$, for $0 \leq i < M$ and $0 \leq j < N-1$. Each processor P_j also has a column bus with M ports, denoted as $\#S_{i,j}$ and each port has m -bit bus connection switches denoted as $\#S_{i,j}(k)$, for $0 \leq i < M$ and $0 \leq k < m$. The m -bit column bus of a processor can be connected to any row bus by setting the port connection switches $\#S_{i,j}(k)$ to $-S_{i,j}(k)$ and/or $+S_{i,j}(k)$ for $0 \leq i < M, 0 \leq j < N$ and $0 \leq k < m$. Assume $M = N = 4$ and $m = 2$, Figure 1 shows a linear RAPWBN.

For a unit of time, assume each processor can either perform one arithmetic or logic operation, or access a local memory word, or set the local switches with the same connection configuration on the same column bus, or communicate with others by broadcasting data on a bus. It allows multiple processors to broadcast data on the different buses or to broadcast the same data on the same bus simultaneously at a time unit, if there is no collision.

Any configuration of the bus system can be derived by properly establishing the local connection among the data bus of each port within each processor. We use the notations as used in [8]. For example, in a linear RAPWBN, if the local connection of a processor is $\{-S_{i,j}(k), +S_{i,j}((k+1) \bmod m), i = 0, 2, 4, \text{ and } 0 \leq k < m\}$ then the m -bit data are rotated one bit after passing through these three switches at the j^{th} column. Figure 2 shows some interesting switch configurations derivable from a processor of a RAPWBN. For simplicity, instead of using $\{-S_{i,j}(k), +S_{i,j}(k), \#S_{i,j}(k), 0 \leq k < m\}$ notation to connect each bit one by one of the i^{th} row and j^{th} column bus network, we use $\{-S_{i,j}, +S_{i,j}, \#S_{i,j}\}$ notation.

A RAPWBN is operated in a single instruction stream, multiple data streams (SIMD) model. Usually, the bus bandwidth is not unlimited between processors. We assume the bus bandwidth is bounded by m -bit so that an m -bit data can be transferred between processors in constant time, where m is an integer. The I/O loading (down load and up load) time is fully dependent on how complex the I/O interface between processors

and peripherals will be. It is difficult to estimate accurately how much I/O time should be included to the time complexity of an algorithm. Therefore, the time complexity of an algorithm is assumed to be the sum of the maximal computation time among all processors and the communication time among all processors. This assumption was also used by many researchers [9, 10, 14, 15, 17, 21, 23, 24].

3 Basic Operations

Let B be a binary sequence of size N , where $B = \{b_0, b_1, \dots, b_{N-1}\}$ and b_i is 0 or 1, $0 \leq i \leq N-1$. The prefix sum for the bit one of a binary sequence, ps_j , is defined as

$$ps_j = \sum_{i=0}^j b_i, \quad (1)$$

where $b_i = 1$, $b_j = 1$, and $0 \leq j \leq N-1$. For example, assume $B = \{1, 1, 0, 1, 0, 1, 1, 1\}$. Then $ps_0 = 1$, $ps_1 = 2$, $ps_3 = 3$, $ps_5 = 4$, $ps_6 = 5$, and $ps_7 = 6$. From Eq. (1), the maximum prefix sum of ps_j is at most N . The prefix sum problem has been solved in RAPWBN by Lee et al. [8], we include it as follows.

Lemma 1 [8] *Given a binary sequence of length N , the prefix sum for the bit one of it can be computed in $O(1)$ time on a linear N RAPWBN each bus with $N^{1/c}$ -bit bandwidth, where c is a constant and $c \geq 1$.*

Let $psum_j$ be the prefix sum of N 's $O(\log N)$ -bit integers and it is defined as

$$psum_j = \sum_{i=0}^j A_i, \quad (2)$$

where $0 \leq A_i < N$ and $0 \leq j < N$. Based on the prefix sum of a binary sequence, Olariu et al. [18] proposed an $O(1)$ time algorithm for this problem on reconfigurable meshes using $N \times N$ processors. Kao et al. [4] also solved this problem on a linear RAP using $N^{1+1/c}$ processors with $N^{1/c}$ -bit bus width, where c is a constant and $c \geq 1$. Kao also extended this result to real numbers. Assume a real number is represented by a normalized floating-point representation which consists of two parts, mantissa and exponent. The result for the integer number can be extended to the real number by following steps. First, find the maximum exponent part from these N real numbers. Next, adjust all real numbers with the maximum exponent part. Then, compute the prefix sum of these N mantissa. Finally, normalize these N prefix sums. We list Kao's result as follows.

Lemma 2 [4] *Given N 's $O(\log N)$ -bit normalized real numbers, the prefix sum of these N real numbers can be computed in $O(1)$ time on a linear $N^{1+1/c}$ RAP with $N^{1/c}$ -bit bus width, where c is a constant and $c \geq 1$.*

Let A_0, A_1, \dots, A_{N-2} and A_{N-1} be N 's $\log N$ -bit unsigned integers, where $0 \leq A_i \leq N-1$ and $0 \leq i \leq N-1$. The maximum (minimum) operation of these N numbers is to determine if there is a number which is not less (greater) than the others. Based on the base- m number system and the prune-and-search technique, the algorithm for finding the maximum of N 's $\log N$ -bit unsigned integers was proposed by Kao et al. [3]. Assume all numbers are distinct and each A_i is represented by the base-2 number system, where

$$A_i = \sum_{j=0}^{R-1} b_{i,j} 2^j, \quad (3)$$

for $R = \lfloor \log_2 N \rfloor + 1$, $b_{i,j} \in \{0, 1\}$, $0 \leq A_i \leq N-1$, $0 \leq i \leq N-1$ and $0 \leq j \leq R-1$. Instead of using the base-2 number system to represent A_i , A_i can be represented by the base- m number system as follows.

$$A_i = \sum_{k=0}^{T-1} a_{i,k} m^k, \quad (4)$$

for $T = \lfloor \log_m N \rfloor + 1$, $0 \leq i \leq N-1$, $0 \leq k \leq T-1$ and $0 \leq a_{i,k} \leq m-1$. From above equation, each A_i is represented by T digits and each digit is bounded within the interval $[0, m-1]$. The maximum (minimum) number of these N unsigned integers can be found using the prune-and-search technique; for each digit $a_{i,k}$ of A_i , if $a_{i,k}$ is greater (less) than $a_{j,k}$ then A_j is pruned. This process repeats from the most significant digit to the least significant digit. We list Kao's result as follows.

Lemma 3 [3] *The maximum (minimum) of N 's $\log N$ -bit unsigned integers can be computed in $O(1)$ time on a linear N RAP each bus with $N^{1/c}$ -bit bandwidth, where c is a constant and $c \geq 1$.*

The above two lemmas are also true in the RAP-WBN model, as we can use a RAPWBN with one-row and N -column bus networks to simulate the RAP.

4 Histogram

The computation of the histogram is a basic operation in image processing and computer vision. It can be used for segmentation and measuring the textual properties of an image. The histogram computation

has been studied extensively by several researchers using various computation models [8, 10, 11, 18].

In this paper, we use the algorithm proposed by Lee et al. [8] for the histogram computation. For the sake of completeness, we include their algorithm as follows. Assume the range of the gray-level of each pixel is bounded by $[0, G-1]$, where G is any constant. The histogram of an image with $n \times n$ pixels can be represented by a one dimension array $H[k]$, $k = 0, 1, \dots, G-1$, such that $H[k] = l$ if l is the number of pixels with gray-level k . Following Lemma 1, the number of each $H[i]$ can be computed by the i^{th} row bus and its corresponding allocated processors. After identifying the last processor that holds gray-level i , we store the histogram of the image pixels each with gray-level i back to processor i . The detailed histogram algorithm is described as follows. Assume the RAPWBN consists of n^2 PE's and each bus has m -bit bandwidth. Initially, each pixel of the $n \times n$ image with gray-level $g_{x,y}$, $0 \leq x, y < n$ and $0 \leq g_{x,y} < G$, is stored in the $gl(j)$ local variable of processor P_j , where $j = x \times n + y$, by row major order. Finally the histogram array $H[j]$, $j = 0, 1, \dots, G-1$, of an $n \times n$ image is stored in the $h(j)$ local variable of processor P_j for all $0 \leq j < G$.

Algorithm HISTOGRAM

Input : gl

Output: h

1. // Set the local connection of each row bus. //
Each processor P_j sets its local connection $\{-S_{i,j}, +S_{i,j}\}$, where $0 \leq i < G$, $0 \leq j < n^2$. Then, for each processor P_j , $0 \leq j < n^2$, establish the local connection $\{-S_{i,j}(k), +S_{i,j}((k+1) \bmod m), \#S_{i,j}\}$, $0 \leq k < m$, if $i = gl(j)$ for $0 \leq i < G$.
2. // Compute the histogram. //
Use the i^{th} row bus to count up the number of pixels $h(i)$ that has grey-level $gl(j) = i$ for each processor P_j , $0 \leq j < n^2$, $0 \leq i < G$. By Lemma 1, at the i^{th} row bus, the processor whose allocated image pixel has grey-level i will be contributed to the computation only. Hence, the histogram can be computed by Lemma 1 for each row bus simultaneously.
3. // Identify the last processor that has the grey-level i for the i^{th} row bus. //
Set the local connection $\{-S_{i,j}, +S_{i,j}\}$ for each processor P_j , $0 \leq j < n^2$, where $0 \leq i < G$, if $i \neq gl(j)$; set the local connection $\{+S_{i,j}, \#S_{i,j}\}$, otherwise. Then the processor P_{n^2-1} broadcasts a signal '*' (or any signal) by the bit 0 of port $+S_{i,n^2-1}$ (i.e. $+S_{i,n^2-1}(0)$) through the established bus. The processor which receives the signal

'*' on the i^{th} row bus is the processor that has the grey-level i . By Step 2, the result $h(i)$ is stored in this processor.

4. // Store $h(i)$ back to processor P_i . //
The processor, which is stored the $h(i)$, copies it back to processor P_i through the i^{th} row bus.

We also have the following theorem.

Theorem 1 [8] For an $n \times n$ image, the histogram can be computed in constant time on a linear n^2 RAPWBN each bus with $(n^2)^{1/c}$ -bit, where c is a constant and $c \geq 1$.

5 Segmentation based on the histogram entropy

The most important thing regarding segmentation is to choose a gray-level as a threshold which divides the N gray-level image into a bi-level image. Let $G_L = 0, 1, \dots, G-1$ be the set of gray levels and $F = [f(x, y)]_{n \times n}$ be an image of size $n^2 = N$, where $f(x, y)$ is the gray-level at (x, y) and $f(x, y) \in G_L$. Let N_i be the number of pixels each with the gray-level i . Then,

$$\sum_{i=0}^{G-1} N_i = N. \quad (5)$$

Pun [19, 20] and Kapur et al. [7] defined the entropy of an image (histogram) as

$$H = - \sum_{i=0}^{G-1} p_i \log p_i, \quad (6)$$

where $p_i = N_i/N$. We use the concept as used by Kapur et al. [7] to define the entropy of the object (black) and the background (white) as follows. The entropy of the black portion of an image is

$$H_B(s) = - \sum_{i=0}^s \frac{p_i}{P_s} \log \frac{p_i}{P_s}; \quad (7)$$

and the entropy of the white portion of the image is

$$H_W(s) = - \sum_{i=s+1}^{G-1} \frac{p_i}{1-P_s} \log \frac{p_i}{1-P_s}, \quad (8)$$

where s is a threshold and $P_s = \sum_{i=0}^s p_i$. The total entropy of the partitioned image is

$$H_T(s) = H_B(s) + H_W(s). \quad (9)$$

Then, for all s , $0 \leq s \leq G - 1$, the s that maximizes the $H_T(s)$ is the threshold value of the image.

Instead of using a linear RAPWBN for the computations, we can configure it into a two dimensional RAPWBN through the following mapping technique. First, reset (disconnect) all switches. Second, processor P_i sets its local connection $\{+S_{0,i}, \#S_{0,i}\}$, if $i \bmod n = 0$; sets its local connection $\{-S_{0,i}, \#S_{0,i}\}$, if $i \bmod n = n-1$; sets its local connection $\{-S_{0,i}, +S_{0,i}, \#S_{0,i}\}$, otherwise. Third, each processor P_i , $i \bmod n = j$ sets its local connection $\{-S_{j+1,i}, +S_{j+1,i}, \#S_{j+1,i}\}$. For the purpose of differentiation, the switches in bus-0 of all processors are called *row-switch* and those in other buses are called *column-switch*. After the reconfiguration, the switch setting is shown in Figure 3(a) for $n = 3$; by this way, the one-dimensional array of processors can be viewed as a two dimensional array of processors, as shown in Figure 3(b). Assume each processor has three variables to keep its indices, i , x and y , where $x = \frac{i}{n}$, $y = i \bmod n$; i is used to represent the index of a processor in a one-dimensional array of processors and x and y are used to represent the index of the same processor in a corresponding two-dimensional array of processors. The mapping between i and x and y are one-to-one and onto.

For calculating the $H_B(s)$, $H_W(s)$ and $H_T(s)$, we arrange the processors to compute these entropies as shown in Figure 4. That is, $H_B(s)$ is computed in the lower triangular processors and $H_W(s)$ is computed in the upper triangular processors. Assume I is a 4×4 image and $I = \{(0, 0, 0, 0), (1, 3, 3, 2), (2, 3, 3, 1), (0, 0, 0, 0)\}$. A snapshot of algorithm SEGMENTATION is shown in Figure 5. In algorithm SEGMENTATION, we assume the number of processors is $n^{2+\frac{1}{2}}$ and the number of gray levels G is n .

Algorithm SEGMENTATION

Input : The gray-level gl of each pixel.

Output: A binary image.

1. // Compute the histogram. //
By algorithm HISTOGRAM, we can compute the histogram in a linear RAPWBN. Assume the final results $H(0), H(1), \dots, H(G-2)$ and $H(G-1)$ are stored in $h(0, 0), h(0, 1), \dots, h(0, G-2)$ and $h(0, G-1)$ local variables of processor $P_{0,0}, P_{0,1}, \dots, P_{0,G-2}$ and $P_{0,G-1}$, respectively.
2. // Reconfigure the architecture. //
Map the one-dimensional array of processors of size $n^{2+\frac{1}{2}}$ to the two-dimensional array of processors of size $n \times n^{1+\frac{1}{2}}$ through the mapping technique. See Figure 3.
3. // Calculate the probability p_i of each gray-level.

//
Each processor $P_{0,y}$, $0 \leq y < G-1$ calculates the probability $p_i(0, y) = \frac{h(0,y)}{N}$, where $p_i(0, y)$ is the local variable of processor $P_{0,y}$.

4. // Compute the prefix sum of the probability p_i . //
According to Lemma 2, the prefix sum of $p_i(0, y)$, $0 \leq y \leq G-1$ can be computed on processors $P_{0,i}$, $0 \leq i \leq n-1$. The sum of each prefix is stored in the local variables $p_s(0, y)$ of processor $P_{0,y}$, $0 \leq y \leq G-1$.
5. // Broadcast the p_s . //
Each processor $P_{0,y}$, $0 \leq y < G-1$ sends its p_s to processor $P_{y,y}$. Then, processor $P_{y,y}$, $0 \leq y \leq G-1$, broadcasts p_s to processor $P_{y,k}$, $0 \leq k \leq G-1$.
6. // Broadcast the p_i . //
Processor $P_{0,y}$ broadcasts $p_i(0, y)$ to processor $P_{x,y}$, $0 \leq x, y \leq G-1$.
7. // Compute the individual entropy. //
Processor $P_{x,y}$, $0 \leq x, y \leq G-1$ computes the entropy $\frac{p_i(x,y)}{p_s(x,y)} \log \frac{p_i(x,y)}{p_s(x,y)}$ and stores it in its local variable $hb(x, y)$, if $x \geq y$; computes the entropy $\frac{p_i(x,y)}{(1-p_s(x,y))} \log \frac{p_i(x,y)}{(1-p_s(x,y))}$ and stores it in its local variable $hw(x, y)$, otherwise.
8. // Calculate the total entropy H_T in each row. //
Compute the total entropy $H_T(x)$ for each x , $0 \leq x \leq G-1$, by the prefix sum on the entropy stored in each row using Lemma 2. Let $H_T(x)$ be stored in the local variable $ht(x, G-1)$ of processor $P_{x,G-1}$. Then set $ht(x, G-1)$ to $-ht(x, G-1)$ in processor $P_{x,G-1}$.
9. // Find the maximum entropy among all $H_T(x)$, $0 \leq x \leq G-1$. //
 - a. Processor $P_{x,G-1}$ broadcasts $ht(x, G-1)$ to processor $P_{x,k}$, $0 \leq k \leq G-1$ and $P_{x,k}$ stores it in local variable $ht_1(x, k)$.
 - b. Processor $P_{x,x}$ broadcasts $ht_1(x, x)$ to processor $P_{k,x}$, $0 \leq k \leq G-1$ and $P_{k,x}$ stores it in local variable $ht_2(k, x)$.
 - c. Processor $P_{x,y}$, $0 \leq x, y \leq G-1$, compares $ht_1(x, y)$ and $ht_2(x, y)$ and sets its $flag(x, y) = 1$, if $ht_1(x, y) \geq ht_2(x, y)$; sets $flag(x, y) = 0$, otherwise.
 - d. Using Lemma 1, we can compute the prefix sum on the $flag$ stored in each row. Processor $P_{x,G-1}$ has the total sum of $flag$ stored in each row and if the total sum of $flag$ is equal to $G-1$ then this processor broadcasts

its index x (the threshold value) to all processors.

10. // Obtain the binary image. // Processor $P_{x,y}$, $0 \leq x, y \leq n - 1$, compares its gray-level value $gl(x, y)$ with the threshold value and sets its gray-level to be one, if $gl(x, y)$ is greater than or equal to the threshold value; sets its gray-level to be zero, otherwise.

Theorem 2 *Let the size of an image be $n \times n$ and the number of gray levels be n . Based on the histogram entropy of the image, the RAPWBN of size $n^{2+\frac{1}{c}}$ can correctly compute the segmentation in constant time.*

Proof: The number of pixels with the same gray-level i , N_i , is computed in Step 1. Then, the probability of each gray-level i , p_i , is calculated in Step 3. The prefix sum of the probability p_i , p_s , is computed in Step 4. The individual entropy of Eqs.(7) and (8) is calculated in Step 7. Since $H_B(s)$ is computed within the range 0 to s and $H_W(s)$ is computed within the range $s + 1$ to $G - 1$, the total entropy of Eq.(9) for each s can be computed by the s^{th} row in Step 8. Then, the threshold value is found in Step 9. Finally, the binary image is obtained by Step 10. The time complexity is analyzed as follows. By Lemma 2, Step 2 and Step 8 each takes $O(1)$ time using $N^{1+\frac{1}{c}}$ processors for each row. Other steps each also takes $O(1)$ time. Hence, the total time complexity is $O(1)$. Q.E.D.

If the linear RAPWBN with n^2 processors is configured to $n^{\frac{2c}{2c+1}} \times n^{\frac{2c+2}{2c+1}}$ two dimensional array, we can compute the segmentation with gray-level $G \leq n^{\frac{2c}{2c+1}}$. This leads to the following corollary.

Corollary 1 *Let the size of an image be $n \times n$ and the number of gray levels be $n^{\frac{2c}{2c+1}}$. Based on the histogram entropy of the image, the RAPWBN of size n^2 can correctly compute the segmentation in constant time.*

6 Image labeling

After segmentation, we get a binary image which has only black and white pixels. Let the black pixels become a group, if they are neighbors (by four-neighbor definition). Then, we give each group a unique label. The labeling algorithm is described as following.

Algorithm LABELLING

Input : A binary image.

Output: A binary image with its black pixels labelled.

1. // Variable initialization. // Processor $P_{x,y}$, $0 \leq x, y \leq n - 1$ sets the local variable $flag(x, y)$ to zero.

2. // Construct each group. // Processor $P_{x,y}$ with a black pixel checks the gray-level of its four neighbors and then breaks the connection with the processor which has the white pixel by setting either *row-switch* or *column-switch* disconnected.
3. // Find the representative of each group. // In each group, we can find the processor with the maximum index i by using Lemma 3. Then, the *flag* of this processor (the representative of the corresponding group) is set to be "1".
4. // Find the label of each group. // Compute the prefix sum on the *flag* stored in each processor. Following Lemma 1, each processor with *flag* = 1 obtains a unique number (i.e. label) to represent its corresponding group.
5. // Broadcast the label by the representative processor. // Reconstruct each group as stated in Step 2. Then, the processor who is the representative of its corresponding group broadcasts the label to all members of its group.

Theorem 3 *For an $n \times n$ binary image, the image labelling problem can be solved in constant time on a linear n^2 RAPWBN each bus with $(n^2)^{1/c}$ -bit, where c is a constant and $c \geq 1$.*

Proof: The correctness of this algorithm can be verified by Lemma 3 and Lemma 1. By Lemma 3, it identifies the representative of each group. By Lemma 1, each group is labelled uniquely. The time complexity can be easily verified to be $O(1)$. Q.E.D.

7 Conclusion

The system bus bandwidth determines the capacity of data communication between processors. According to the results as shown in [10, 13], we know that the silicon area used by the switching control mechanism is far less than that used by the processor. Instead of increasing the number of processors, we extend the number of buses to increase the power of a parallel processing system. Such a strategy of utilizing the reconfigurable array of processors with wider bus networks not only has the advantage of saving silicon but also increases the system power enormously.

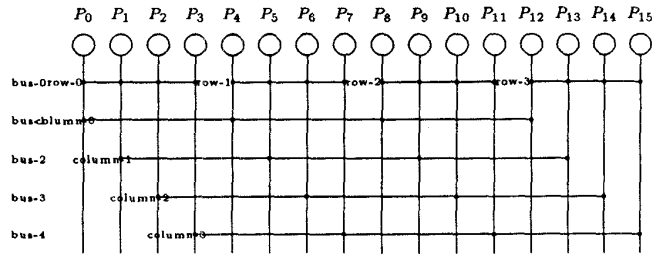
In this paper, to demonstrate the power of the RAPWBN, three related image problems such as histogram, segmentation and labelling are proposed for this machine. All problems can be solved in constant time.

The RAPWBN is designed to be suitable for image processing. We believe that lots of image related problems can be solved in constant time in the near future.

References

- [1] H. M. Alnuweiri, "Fast algorithms for image labeling on reconfigurable network of processors," *International Parallel Processing Symposium*, pp. 569-575, 1993.
- [2] L. Cinque, S. Levialdi and A. Rosenfeld, "Fast pyramidal algorithms for image thresholding," *Pattern Recognition*, vol. 28, no. 6, pp. 901-906, 1995.
- [3] T. W. Kao and S. J. Horng, "Parallel computing two nearest neighbor problems on a RAP", *Seventeenth Annual Computer Science Conference*, pp. 135-144, Jan. 1994.
- [4] T. W. Kao and S. J. Horng, "Computing list ranking on a RAP with wider bus networks", *Proceeding of the 1994 International Conference on Parallel and Distributed Systems*, pp. 28-33, Dec. 1994.
- [5] T. W. Kao and S. J. Horng, "Efficient parallel algorithms for image processing on CRAP", *Technical Report*, Department of Electrical Engineering, National Taiwan Institute of Technology, 1992.
- [6] T. W. Kao, S. J. Horng and H. R. Tsai, "Computing connected components and some related applications on a RAP," *Proc. Int. Conf. Parallel Proc.*, vol. 3, pp. 57-64, 1993.
- [7] J. N. Kapur, P. K. Sahoo and A. K. C. Wong, "A new method for gray level picture thresholding using the entropy of the histogram," *Computer Graphics, Vision and Image Processing*, vol. 29, pp. 273-285, 1985.
- [8] S. S. Lee, S. J. Horng, T. W. Kao and H. R. Tsai, "Optimal computing Hough transform on a reconfigurable array of processors with wider bus network," to appear in *Pattern Recognition*, 1995.
- [9] H. Li and M. Maresca, "Polymorphic-torus network," *IEEE Transactions on Computers*, vol. 38, no. 9, pp. 1345-1351, Sep. 1989.
- [10] H. Li and M. Maresca, "Polymorphic-torus architecture for computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 3, pp. 233-243, Mar. 1989.
- [11] W. M. Lin and V. K. P. Kumar, "Efficient histogramming on hypercube SIMD machines," *Computer Vision, Graphics, and Image Processing*, vol. 49, pp. 104-120, 1992.
- [12] M. Maresca, "Polymorphic processor arrays," *IEEE Trans. Para. and Distr. Sys.*, 4, pp. 490-506, 1993.
- [13] M. Maresca and H. Li, "Connection autonomy in SIMD computers: a VLSI implementation," *Journal of Parallel and Distributed Computing*, vol. 7, no. 2, pp. 302-320, 1989.
- [14] R. Miller, V. K. P. Kumar, D. Reisis and Q. F. Stout, "Meshes with reconfigurable buses," *Proceedings of the MIT Conference on Advanced Research in VLSI*, pp. 163-178, Mar. 1988.
- [15] R. Miller, V. K. P. Kumar, D. Reisis and Q. F. Stout, "Data movement operations and applications on reconfigurable VLSI arrays," *Proceedings of the International Conference on Parallel Processing*, vol. 1, pp. 205-208, Aug. 1988.
- [16] D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," *IEEE Trans. Comput.*, vol. C-30, pp. 101-107, Feb. 1981.
- [17] S. Olariu, J. L. Schwing and J. Zhang, "Fundamental data movement algorithms for reconfigurable mesh," *Proc. 11-th Annual International Phoenix Conference on Computers and Communications*, Scottsdale, Arizona, pp. 0480-0484, April, 1992.
- [18] S. Olariu, J. L. Schwing and J. Zhang, "Fast computer vision algorithms for reconfigurable mesh," *Image and Vision Computing*, vol. 10, no. 9, pp. 610-616, Nov. 1992.
- [19] T. Pun, "A new method for gray-level picture thresholding using the entropy of the histogram," *Signal Processing*, vol. 2, pp. 223-237, 1980.
- [20] T. Pun, "Entropic thresholding: a new approach," *Signal Processing*, vol. 2, pp. 210-239, 1981.
- [21] J. Rothstein, "Bus automata, brains, and mental models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 4, pp. 522-531, Apr. 1988.

- [22] A Schuster, "Dynamic reconfiguring networks for parallel computers: algorithms and complexity bounds," Doctoral Dissertation, Dept. of Computer Science, Hebrew University, Israel, 1991.
- [23] B. F. Wang, G. H. Chen and F. C. Lin, "Constant time sorting on a processor array with a reconfigurable bus system," *Information Processing Letters*, vol. 34, no. 4, pp. 187-192, Apr. 1990.
- [24] B. F. Wang, G. H. Chen and H. Li, "Configurational computation: a new computation method on processor arrays with reconfigurable bus system," *Proceedings of the International Conference on Parallel Processing*, pp. III-42-III-49, Aug. 1991.



(a) The switch setting for a two-dimensional array.

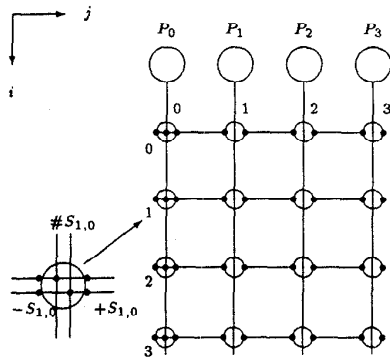
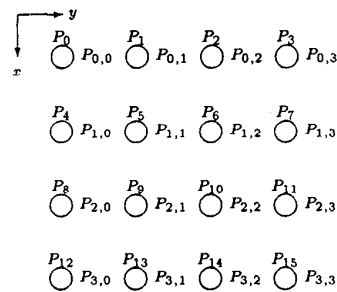


Figure 1: A linear RAPWBN of size 4 with 4×4 bus networks, each bus network with 2-bit bus width.



(b) The emulated two-dimensional array of processors.

Figure 3: The mapping between a one-dimensional array of processors and a two-dimensional array of processors.

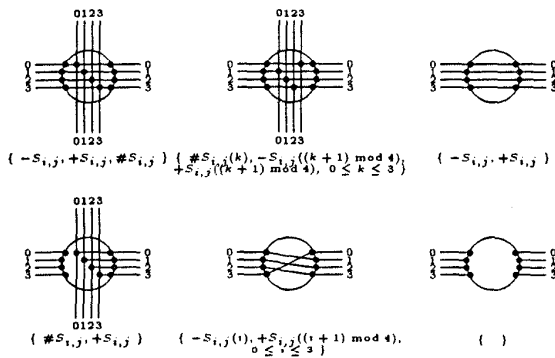


Figure 2: Some local switch configurations of a RAPWBN.

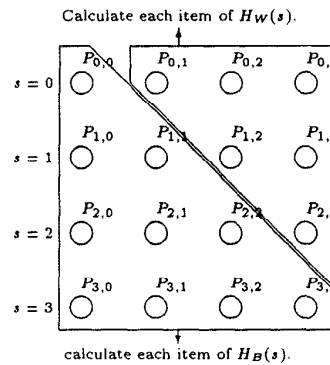


Figure 4: The arrangement of the $H_T(s)$ calculation on processors.